KATARZYNA ŻMUDA

SQL

JAK OSIĄGNĄĆ MISTRZOSTWO W KONSTRUOWANIU ZAPYTAŃ

Helion

Redaktor prowadzący: Michał Mrowiec

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/sqljak
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Kody źródłowe wybranych przykładów dostępne są pod adresem:
ftp://ftp.helion.pl/przyklady/sqljak.zip

ISBN: 978-83-283-1283-8

Printed in Poland.

Spis treści

Rozdział 4.

Funkcje agregujące

Funkcje agregujące są to funkcje, które jako parametr przyjmują zbiór wartości i zwracają w wyniku pojedynczą wartość. Ten parametr podajemy zazwyczaj w postaci nazwy kolumny, a obliczenia są wykonywane na wszystkich wartościach występujących w tej kolumnie.

W tym rozdziale omówimy następujące funkcje agregujące:

- `COUNT` — zlicza liczbę wierszy,
- `SUM` — oblicza sumę wartości,
- `MIN` — znajduje najniższą wartość,
- `MAX` — znajduje najwyższą wartość,
- `AVG` — oblicza średnią wartość.

Ćwiczenia

Ćwiczenia będą wykonywane na tych samych tabelach `Employee`, `Department` i `City`, co ćwiczenia z rozdziału 3.

Ćwiczenie 4.1

Policz wszystkich pracowników.

Oczekiwany wynik

Rysunek 4.1.
Oczekiwany wynik ćwiczenia 4.1

LiczbaPracowników
10

Rozwiązanie

Aby policzyć wszystkich pracowników, skorzystamy z funkcji agregującej `COUNT`, którą wywołamy z użyciem parametru *:

```
SELECT COUNT(*) as LiczbaPracowników
FROM Employee
```

Funkcja `COUNT` wywołana z użyciem parametru * zlicza wszystkie wiersze w danej tabeli.

Jeśli natomiast jako parametr podamy nazwę kolumny, zostaną zliczone wszystkie wartości w tej kolumnie, które nie są wartościami `NULL`.

Jako przykład weźmy tabelę 4.1 o nazwie `Słownik`.

Tabela 4.1. *Słownik*

Id	Nazwa
1	AAA
3	BBB
4	NULL
5	CCC

Dla zapytania:

```
SELECT COUNT(*)
FROM Słownik
```

otrzymamy wynik `4` —zostały zliczone wszystkie wiersze.

Dla zapytania:

```
SELECT COUNT(Nazwa)
FROM Słownik
```

otrzymamy wynik `3` — zostały zliczone te wiersze, które w kolumnie `Nazwa` mają wartości inne niż `NULL`.

Ćwiczenie 4.2

Policz wszystkich pracowników, którzy nie mają drugiego imienia.

Oczekiwany wynik

Rysunek 4.2. *Oczekiwany wynik ćwiczenia 4.2*

Rozwiązanie

Aby znaleźć pracowników, którzy nie mają drugiego imienia, należy wyświetlić te wszystkie wiersze z tabeli `Employee`, których wartość w kolumnie `SecondName` jest wartością `NULL`. Następnie należy — za pomocą funkcji agregującej `COUNT` — policzyć liczbę zwróconych wierszy:

```
SELECT COUNT(*) as LiczbaPracownikow
FROM Employee
WHERE SecondName IS NOT NULL
```

To zadanie można rozwiązać również w inny sposób.

Tak jak pokazałam w poprzednim ćwiczeniu — jeżeli funkcja `COUNT` jako parametr przyjmuje nazwę kolumny, wówczas zlicza tylko te wartości z tej kolumny, które nie są wartościami `NULL`.

Powyższe zapytanie można więc również zapisać w następujący sposób:

```
SELECT COUNT(SecondName)
FROM Employee
```

Ćwiczenie 4.3

Policz wszystkich aktywnych pracowników.

Oczekiwany wynik

Rysunek 4.3. *Oczekiwany wynik ćwiczenia 4.3*

LiczbaAktywnychPracowników
7

Rozwiązanie

Jeśli używamy funkcji agregujących, nic nie stoi na przeszkodzie, aby używać również poznanych już wcześniej elementów języka — w tym wypadku warunków, których użyjemy w sekcji `WHERE`.

Aby policzyć wszystkich aktywnych pracowników, użyjemy funkcji agregującej `COUNT(*)` dla wierszy, które spełniają warunek `Active = 1`.

Zapytanie więc będzie wyglądało następująco:

```
SELECT COUNT(*) as LiczbaAktywnychPracowników
FROM Employee
WHERE Active = 1
```

Ćwiczenie 4.4

Policz, ilu pracowników pracuje w dziale IT.

Oczekiwany wynik

Rysunek 4.4. *Oczekiwany wynik ćwiczenia 4.4*

Rozwiązanie

Informacje o pracownikach mamy w tabeli `Employee`, natomiast informacje o dziale, w którym pracują, mamy w tabeli `Department`. Musimy więc złączyć te dwie tabele oraz zdefiniować warunek tak, aby zostały wyświetlone dane tylko tych pracowników, którzy są zatrudnieni w dziale IT:

```
SELECT *
FROM Employee
INNER JOIN Department ON Employee.DepartmentId = Department.Id
WHERE Department.Name = 'IT'
```

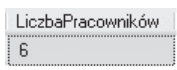
Teraz wystarczy już tylko zliczyć wiersze za pomocą funkcji agregującej `COUNT(*)`:

```
SELECT COUNT(*) as LiczbaPracowników
FROM Employee
INNER JOIN Department ON Employee.DepartmentId = Department.Id
WHERE Department.Name = 'IT'
```

Ćwiczenie 4.5

Policz wszystkich pracowników, którzy pracują w działach znajdujących się w Warszawie.

Oczekiwany wynik



Rysunek 4.5. *Oczekiwany wynik ćwiczenia 4.5*

Rozwiązanie

Informacje o pracownikach znajdują się w tabeli `Employee`, dane działów — w tabeli `Department`, dane miast — w tabeli `City`. Musimy więc połączyć te trzy tabele. Następnie definiujemy warunek wyświetlający tylko wiersze, w których `Name` ma wartość `Warsaw`, i zliczamy wiersze za pomocą funkcji agregującej `COUNT`:

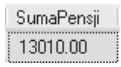
```
SELECT COUNT(*) as LiczbaPracownikow
FROM Employee
INNER JOIN Department ON Employee.DepartmentId = Department.Id
INNER JOIN City ON Department.CityId = City.Id
WHERE City.Name = 'Warsaw'
```

Ćwiczenie 4.6

Policz, ile pracodawca wydaje miesięcznie na pensje. Zauważ, że pracodawca płaci pensje tylko aktywnym pracownikom.

Oczekiwany wynik

Rysunek 4.6. *Oczekiwany wynik ćwiczenia 4.6*



Rozwiązanie

Żeby policzyć, ile pracodawca wydaje miesięcznie na pensje, musimy policzyć sumę zarobków wszystkich aktywnych pracowników.

Aktywnych pracowników znajdziemy, używając zapytania:

```
SELECT *
FROM Employee
WHERE Active = 1
```

Aby obliczyć sumę, skorzystamy z funkcji agregującej `SUM`, która jako parametr przyjmuje nazwę kolumny i zlicza sumę wartości z tej kolumny. Aby wyświetlić sumę zarobków, użyjemy funkcji `SUM` w sekcji `SELECT`, zamiast *:

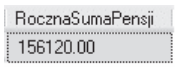
```
SELECT SUM(Salary) as SumaPensji
FROM Employee
WHERE Active = 1
```

Ćwiczenie 4.7

Policz, ile pracodawca wydaje rocznie na pensje.

Oczekiwany wynik

Rysunek 4.7. *Oczekiwany wynik ćwiczenia 4.7*



Rozwiązanie

Aby policzyć roczną sumę zarobków, należy pomnożyć miesięczne zarobki przez `12`.

Zapytanie będzie więc wyglądało następująco:

```
SELECT SUM(Salary) * 12 as RocznaSumaPensji
FROM Employee
WHERE Active = 1
```

Można je również zapisać w taki sposób:

```
SELECT SUM(Salary * 12) as RocznaSumaPensji
FROM Employee
WHERE Active = 1
```

Ćwiczenie 4.8

Znajdź minimalne zarobki spośród wszystkich pracowników.

Oczekiwany wynik

Rysunek 4.8.
Oczekiwany wynik ćwiczenia 4.8

Rozwiązanie

Aby znaleźć minimalne zarobki, użyjemy funkcji `MIN`, która jako parametr przyjmuje nazwę kolumny i zwraca najmniejszą wartość spośród wszystkich w tej kolumnie.

Zapytanie będzie więc wyglądało następująco:

```
SELECT MIN(Salary) as MinimalneZarobki
FROM Employee
```

Ćwiczenie 4.9

Policz, z ilu liter składa się najkrótsze imię pracownika spośród zatrudnionych.

Oczekiwany wynik

Rysunek 4.9.
Oczekiwany wynik ćwiczenia 4.9

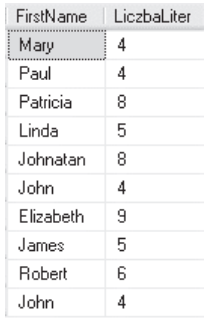
Rozwiązanie

Liczbę liter w imieniu możemy policzyć za pomocą funkcji `LEN`:

```
SELECT FirstName, LEN(FirstName) as LiczbaLiter
FROM Employee
```

Po wykonaniu tego zapytania otrzymamy wynik widoczny na rysunku 4.10.

Rysunek 4.10.
Wynik obliczenia liczby liter w imionach wszystkich pracowników



FirstName	LiczbaLiter
Mary	4
Paul	4
Patricia	8
Linda	5
Johnatan	8
John	4
Elizabeth	9
James	5
Robert	6
John	4

Na rysunku widzimy, że spośród wyświetlonych długości imion najkrótsze z nich ma 4 znaki. Aby napisać zapytanie, które znajdzie najkrótsze imię, musimy użyć funkcji `MIN`, jako parametr podając wynik funkcji `LEN(FirstName)`:

```
SELECT MIN(LEN(FirstName)) as NajmniejLiter
FROM Employee
```

Ćwiczenie 4.10

Policz, ile lat łącznie przepracowali w firmie aktywni pracownicy.

Oczekiwany wynik

Rysunek 4.11.
Oczekiwany wynik ćwiczenia 4.10

SumaLat
18

Rozwiązanie

Aby wyświetlić, ile lat przepracowali firmie aktywni pracownicy, użyjemy funkcji `DATEDIFF` — do policzenia różnicy lat pomiędzy datą zatrudnienia a datą dzisiejszą:

```
SELECT DATEDIFF(year,HireDate,GETDATE())
FROM Employee
WHERE Active = 1
```

Aby obliczyć sumę tych lat, użyjemy funkcji `SUM`, jako argument podając wynik funkcji `DATEDIFF`:

```
SELECT SUM(DATEDIFF(year,HireDate,GETDATE()))
FROM Employee
WHERE Active = 1
```

Ćwiczenie 4.11

Sprawdź, jaka jest różnica pomiędzy minimalną a maksymalną pensją.

Oczekiwany wynik

Rysunek 4.12. *Oczekiwany wynik ćwiczenia 4.11*

RoznicaPensji
1900.00

Rozwiązanie

Aby znaleźć wysokość minimalnej pensji, użyjemy poznanej już funkcji `MIN`.

```
SELECT MIN(Salary)
FROM Employee
```

Aby znaleźć wysokość maksymalnej pensji, użyjemy analogicznej funkcji `MAX`.

```
SELECT MAX(Salary)
FROM Employee
```

Aby policzyć różnicę pomiędzy minimalną a maksymalną wysokością pensji, użyjemy tych dwóch funkcji w jednym zapytaniu i znajdziemy różnicę pomiędzy nimi za pomocą operatora odejmowania:

```
SELECT MAX(Salary) - MIN(Salary) as RoznicaPensji
FROM Employee
```

Ćwiczenie 4.12

Policz średnią, minimalną i maksymalną wysokość pensji pracowników w dziale IT.

Oczekiwany wynik

Rysunek 4.13. *Oczekiwany wynik ćwiczenia 4.12*

AvgSalary	MinSalary	MaxSalary
1952.500000	1200.00	2500.00

Rozwiązanie

Poznaliśmy już funkcje `MIN` i `MAX`, które zwracają odpowiednio najniższą i najwyższą wartość. Istnieje jeszcze jedna funkcja agregująca `AVG`, która zwraca średnią wartość.

Zastosujemy te trzy funkcje w kolumnie `Salary` w wynikach połączonych tabel `Employee` oraz `Department`:

```
SELECT AVG(Salary) as AvgSalary, MIN(Salary) as MinSalary, MAX(Salary) as MaxSalary
FROM Employee
INNER JOIN Department ON Employee.DepartmentId = Department.Id
WHERE Name = 'IT'
```

Ćwiczenie 4.13

O ile należałoby podnieść pensję pracownikowi, który zarabia najmniej, żeby jego pensja wynosiła tyle, ile średnia pensja?

Oczekiwany wynik

Rysunek 4.14. *Oczekiwany wynik ćwiczenia 4.13*

Wyrownanie
891.000000

Rozwiązanie

Pensję pracownika, który zarabia najmniej, znajdziemy za pomocą funkcji `MIN`, a wysokość średniej pensji — za pomocą funkcji `AVG`. Aby dowiedzieć się, o ile należałoby podnieść wysokość tej minimalnej pensji, żeby wynosiła ona tyle, co średnia, należy znaleźć różnicę pomiędzy tymi kwotami:

```
SELECT AVG(Salary) - MIN(Salary)
FROM Employee
```

Ćwiczenie 4.14

Policz średni wiek wszystkich pracowników.

Oczekiwany wynik

Rysunek 4.15. *Oczekiwany wynik ćwiczenia 4.14*

SredniWiek
26

Rozwiązanie

Aby wyświetlić wiek pracowników, należy obliczyć liczbę lat, która minęła od dnia ich urodzin do dnia dzisiejszego:

```
SELECT DATEDIFF(year, BirthDate, GETDATE())
FROM Employee
```

Następnie za pomocą funkcji `AVG` znajdujemy średnią z tych wartości:

```
SELECT AVG(DATEDIFF(year, BirthDate, GETDATE())) as SredniWiek
FROM Employee
```

Ćwiczenie 4.15

Policz średnią liczbę miesięcy przepracowanych przez tych pracowników, którzy już w firmie nie pracują.

Oczekiwany wynik

Rysunek 4.16. *Oczekiwany wynik ćwiczenia 4.15*

SredniaLiczbaMiesiecy
12

Rozwiązanie

Aby znaleźć liczbę miesięcy, które przepracowali pracownicy niezatrudnieni już w firmie, użyjemy funkcji `DATEDIFF` — do obliczenia, ile miesięcy minęło od daty zatrudnienia (`HireDate`) do daty zwolnienia (`RelieveDate`), ale tylko dla tych pracowników, którzy mają wpisaną datę zwolnienia. Następnie za pomocą funkcji `AVG` znajdujemy średnią z tych wartości:

```
SELECT AVG(DATEDIFF(month, HireDate, RelieveDate)) as SredniaLiczbaMiesiecy
FROM Employee
WHERE RelieveDate IS NOT NULL
```

Ćwiczenie 4.16

Znajdź najpóźniejszą datę urodzenia spośród wszystkich pracowników.

Oczekiwany wynik

Rysunek 4.17. *Oczekiwany wynik ćwiczenia 4.16*

MaksymalnaDataUrodzenia
1995-05-28

Rozwiązanie

Funkcji agregujących można używać nie tylko na liczbach, ale również na datach:

```
SELECT MAX(BirthDate) as MaksymalnaDataUrodzenia
FROM Employee
```

Ćwiczenie 4.17

Wyświetl wszystkie unikalne wysokości płac w firmie i dla każdej z tych wartości policz, ilu pracowników zarabia taką właśnie kwotę.

Oczekiwany wynik

Rysunek 4.18. *Oczekiwany wynik ćwiczenia 4.17*

Salary	LiczbaPracownikow
1000.00	2
1200.00	1
1500.00	1
2000.00	2
2110.00	1
2500.00	1
2700.00	1
2900.00	1

Rozwiązanie

Umiemy już stosować funkcje agregujące na wszystkich wierszach zapytania, czyli np. policzyć liczbę wierszy w tabeli. Kolejnym krokiem jest możliwość stosowania funkcji agregujących na grupach rekordów. Pokażę to na prostym przykładzie.

Załóżmy, że mamy tabelę `Samochody`, która jest wypełniona danymi jak w tabeli 4.1.

Tabela 4.1. *Przykładowe dane dla samochodów*

Marka	Rok_produkcji	Model
Toyota	**2013**	Yaris
Toyota	**2014**	Auris
Fiat	**2013**	Bravo
Fiat	**2012**	Punto
Toyota	2013	Corolla

Chcielibyśmy policzyć, ile mamy modeli samochodów dla poszczególnych marek. W tym celu musimy pogrupować je według marki, dodając klauzulę `GROUP BY Marka`. Pogrupowaną tabelę możemy sobie wyobrazić jako tabelę przedstawioną poniżej tabela 4.2.

Tabela 4.2. *Wyniki pogrupowania samochodów według marki*

Marka	Rok_produkcji	Model
Toyota	**2013**	Yaris
	2014	Auris
	2013	Corolla
Fiat	**2013**	Bravo
	2012	Punto

Każda marka jest teraz nagłówkiem grupy. Każda grupa posiada wiersze, które pasują do grupy według kryterium grupowania. Na tych wierszach możemy stosować teraz funkcje grupujące (agregujące), które będą wykonywać działania w ramach grupy.

Tabeli 4.2 nie możemy wyświetlić. Narysowałam ją w celu lepszego zobrazowania tego, w jaki sposób wygląda grupowanie.

Jeśli więc wykonamy zapytanie:

```
SELECT Marka, COUNT(*) as Liczba
FROM Samochody
GROUP BY Marka
```

wówczas otrzymamy wynik zaprezentowany w tabeli 4.3.

Tabela 4.3. *Wynik policzenia liczby modeli samochodów dla każdej marki*

Marka	Liczba
Toyota	3
Fiat	2

Jeśli zastosujemy klauzulę `GROUP BY`, to w sekcji `SELECT` możemy wyświetlać tylko i wyłącznie:

- kolumny, które są kryterium grupowania, tak jak marka samochodu w powyższym przykładzie,
- wyniki funkcji grupujących (agregujących).

Nie możemy już odwoływać się do pozostałych kolumn. Przykładowo, jeśli napisalibyśmy:

```
SELECT Marka, Model
FROM Samochody
GROUP BY Marka
```

to jaki model samochodu miałby być wyświetlony? Yaris, Auris czy Corolla? Nie wiadomo. Grupę utworzyliśmy po to, żeby mieć tylko jeden wiersz dla danej grupy, a więc nie możemy wyświetlić kolumn, które nie zostały użyte do grupowania. Jeśli napiszemy takie zapytanie, w wyniku otrzymamy błąd:

```
Column 'Samochody.Model' is invalid in the select list because it is not contained
in either an aggregate function or the GROUP BY clause.
```

Pamiętaj, że jeśli użyjesz grupowania, w sekcji `SELECT` możesz używać tylko funkcji grupujących (agregujących) oraz kolumn, które wystąpiły w sekcji `GROUP BY`.

Grupę można również utworzyć na podstawie wielu kolumn. Na przykład, jeśli chcemy policzyć, ile modeli danej marki zostało wyprodukowanych w poszczególnych latach, możemy pogrupować dane według marki oraz roku produkcji:

```
GROUP BY Marka, Rok_produkcji
```

Wyobrażamy sobie wtedy wyniki pogrupowania tak, jak to przedstawiono w tabeli 4.4.

Tabela 4.4. *Wyniki pogrupowania samochodów według marki i roku produkcji*

Marka	Rok_produkcji	Model
Toyota	**2013**	Yaris
		Corolla
	2014	Auris
Fiat	**2013**	Bravo
	2012	Punto

Możemy teraz np. policzyć liczbę wierszy w danej grupie.

```
SELECT Marka, Rok_produkcji, COUNT(*) as Liczba
FROM Samochody
GROUP BY Marka, Rok_produkcji
```

Wynik tego zapytania został zaprezentowany w tabeli 4.5.

Tabela 4.5. *Wynik zapytania o liczbę wierszy w danej grupie*

Marka	Rok_produkcji	Liczba
Toyota	**2013**	2
Toyota	**2014**	1
Fiat	**2013**	1
Fiat	2012	1

Nawet jeśli widzimy, że w danej grupie znajduje się tylko jeden wiersz, trzymamy się zasady, że nie wolno nam w sekcji SELECT wyświetlać kolumn, które nie są kryterium grupowania.

Wróćmy więc do ćwiczenia. Mamy wyświetlić wszystkie unikalne wartości zarobków i policzyć, ilu pracowników zarabia taką właśnie kwotę.

Budujemy więc grupę według zarobków:

```
GROUP BY Salary
```

Następnie zliczamy liczbę wierszy w każdej grupie:

```
SELECT Salary, COUNT(*) as LiczbaPracownikow
FROM Employee
GROUP BY Salary
```

Ćwiczenie 4.18

Wyświetl wszystkie unikalne wysokości płac w firmie.

Oczekiwany wynik

Rysunek 4.19.
Oczekiwany wynik ćwiczenia 4.18

Salary
1000.00
1200.00
1500.00
2000.00
2110.00
2500.00
2700.00
2900.00

Rozwiązanie

Umiemy już rozwiązać powyższe ćwiczenie za pomocą klauzuli `DISTINCT`:

```
SELECT DISTINCT Salary
FROM Employee
```

Możemy to jednak również zrobić, używając grupowania:

```
SELECT Salary
FROM Employee
GROUP BY Salary
```

To, że w klauzuli `SELECT` możesz użyć tylko kolumn występujących w sekcji `GROUP BY`, nie oznacza, że musisz ich użyć. Zależy to tylko i wyłącznie od Twoich wymagań i potrzeb. W klauzuli `SELECT` może wystąpić dowolny podzbiór kolumn z sekcji `GROUP BY`, mogą też zostać użyte same funkcje grupujące (agregujące).

Przykłady poprawnego użycia:

```
SELECT X, COUNT(*), SUM(Z)
FROM Tabela
GROUP BY X, Y

SELECT COUNT(*)
FROM Tabela
GROUP BY X, Y, Z

SELECT X, Z
FROM Tabela
GROUP BY X, Y, Z
```

Zawsze jeśli patrzysz na zapytanie grupujące, upewnij się, że w sekcji `SELECT` nie ma kolumn, które nie występują w sekcji `GROUP BY`.

Poświęciłam temu zagadnieniu tak wiele miejsca, ponieważ jest to jeden z najczęściej popełnianych błędów dotyczących grupowania.

Ćwiczenie 4.19

Wyświetl, ilu pracowników pracuje w poszczególnych działach.

Oczekiwany wynik

Rysunek 4.20. *Oczekiwany wynik ćwiczenia 4.19*

Name	LiczbaPracownikow
Accounting	2
HR	2
IT	4
Marketing	2

Rozwiązanie

Informacje o pracownikach mamy w tabeli `Employee`, natomiast informacje o działach — w tabeli `Department`. Musimy więc połączyć te dwie tabele:

```
SELECT *
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Następnie wyniki połączenia tabel musimy pogrupować według nazwy działu i policzyć liczbę wierszy w każdej grupie, czyli liczbę pracowników przypisanych danemu działowi:

```
SELECT Name, COUNT(*) as LiczbaPracownikow
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Ćwiczenie 4.20

Wyświetl, ilu pracowników pracuje w poszczególnych działach. Wyniki rozszerz również o te działy, które nie mają przypisanego żadnego pracownika.

Oczekiwany wynik

Rysunek 4.21. *Oczekiwany wynik ćwiczenia 4.20*

Name	LiczbaPracownikow
Accounting	2
Controlling	0
Data Warehouse	0
HR	2
IT	4
Marketing	2

Rozwiązanie

Aby uwzględnić również działy, które nie mają przypisanego żadnego pracownika, użyjemy złączenia zewnętrznego `LEFT JOIN`. Grupowanie i sposób liczenia pracowników będą dokładnie takie same, jak w poprzednim ćwiczeniu:

```
SELECT Name, COUNT(*) as LiczbaPracownikow
FROM Department
LEFT JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Jeśli wykonamy to zapytanie, okazuje się, że dla działów, które nie mają przypisanego żadnego pracownika, np. `Controlling`, otrzymaliśmy wynik `1` (rysunek 4.22).

Rysunek 4.22. *Wyniki z błędnie zwróconą liczbą pracowników dla działów, w których nikt nie pracuje*

Name	LiczbaPracownikow
Accounting	2
Controlling	1
Data Warehouse	1
HR	2
IT	4
Marketing	2

Dla działu `Controlling` istnieje jeden wiersz zawierający informacje o tym dziale, ale brakuje informacji o pracownikach, ponieważ nie istnieje żaden z nich przypisany temu działowi. Pamiętaj o tym, jak działa funkcja `COUNT`. Jeśli parametrem wywołania tej funkcji jest *, zlicza ona wszystkie wiersze — bez względu na to, czy wiersz ten posiada wartości z obu tabel, czy też został uzupełniony wartościami `NULL` dla złączenia zewnętrznego `LEFT JOIN`. Jeśli parametrem wywołania tej funkcji jest nazwa kolumny — zlicza ona tylko te wiersze, w których wartość tej kolumny jest różna od `NULL`. Możemy wykorzystać to działanie w wypadku powyższego ćwiczenia i jako parametr do funkcji `COUNT` przekazać kolumnę `Employee.Id`:

```
SELECT Name, COUNT(Employee.Id) as LiczbaPracownikow
FROM Department
LEFT JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Ćwiczenie 4.21

Dla każdego działu policz liczbę pracowników zarabiających więcej niż 1500 zł.

Oczekiwany wynik

Rysunek 4.23. *Oczekiwany wynik ćwiczenia 4.21*

Name	LiczbaPracownikow
Accounting	2
IT	3
Marketing	1

Rozwiązanie

Jeśli używamy grupowania, wciąż możemy używać warunków w stosunku do kolumn, które nie występują jako kryterium grupowania. W tym przykładzie będziemy grupować według nazwy działów, a w klauzuli `WHERE` sprawdzimy wysokość zarobków.

Przy przetwarzaniu zapytania najpierw są wykonywane warunki z sekcji `WHERE`, mamy więc wówczas dostępne jeszcze wszystkie kolumny, a dopiero potem jest wykonywane grupowanie.

Sekcja `GROUP BY` znajduje się w zapytaniu po sekcji `WHERE`.

Zapytanie będzie więc wyglądało następująco:

```
SELECT Name, COUNT(*) as LiczbaPracownikow
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
WHERE Salary > 1500
GROUP BY Name
```

Ćwiczenie 4.22

Dla każdego działu wyświetl średnią pensję. Wyniki posortuj według nazwy działu.

Oczekiwany wynik

Rysunek 4.24.
Oczekiwany wynik ćwiczenia 4.22

Name	SredniaPensja
Accounting	2800.000000
HR	1250.000000
IT	1952.500000
Marketing	1500.000000

Rozwiązanie

Aby policzyć średnią pensję w dziale, połączymy tabele `Employee` i `Department`. Wyniki pogrupujemy według nazwy działu i użyjemy funkcji `AVG`, której argumentem będzie kolumna `Salary`, aby policzyć średnią pensję:

```
SELECT Name, AVG(Salary) as SredniaPensja
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
```

Otrzymane wyniki posortujemy za pomocą klauzuli `ORDER BY`. Tę klazulę obowiązują takie same zasady jak klauzulę `SELECT`: możemy tu używać tylko tych kolumn, które były kryterium grupowania, oraz funkcji agregujących. Jeśli będziemy chcieli posortować wyniki według nazwiska pracownika (kolumna `LastName` w tabeli `Employee`), która to kolumna nie jest kryterium grupowania, otrzymamy znany już błąd:

```
Column "Employee.LastName" is invalid in the ORDER BY clause because it is not
contained in either an aggregate function or the GROUP BY clause.
```

Sekcja ORDER BY znajduje się na samym końcu zapytania, po sekcji GROUP BY. Zapytanie będzie więc wyglądało następująco:

```
SELECT Name,  AVG(Salary) as SredniaPensja
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
ORDER BY Name
```

Ćwiczenie 4.23

Dla każdego działu policz, ile pieniędzy idzie miesięcznie na wypłaty (czyli dla aktywnych pracowników). Uwzględnij również działy, które nie mają przypisanego żadnego pracownika.

Oczekiwany wynik

Rysunek 4.25. *Oczekiwany wynik ćwiczenia 4.23*

Name	SumaWynagrodzen
Accounting	5600.00
Controlling	0.00
Data Warehouse	0.00
HR	2500.00
IT	7810.00
Marketing	3000.00

Rozwiązanie

Połączymy tabelę Department z tabelą Employee przy użyciu złączenia zewnętrznego LEFT JOIN, tak aby uwzględnić również działy, które nie mają przypisanego żadnego pracownika. Następnie wyniki pogrupujemy według nazwy działu i dla każdej grupy, czyli dla każdego działu, policzymy sumę wynagrodzeń pracowników tego działu:

```
SELECT Name, SUM(Salary) as SumaWynagrodzen
FROM Department
LEFT JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Po wykonaniu zapytania otrzymamy wynik jak na rysunku 4.26.

Rysunek 4.26. *Wynik obliczenia sumy wynagrodzeń dla każdego działu*

Name	SumaWynagrodzen
Accounting	5600.00
Controlling	NULL
Data Warehouse	NULL
HR	2500.00
IT	7810.00
Marketing	3000.00

Zauważ, że po wykonaniu tego zapytania, dla działów, które nie mają przypisanego żadnego pracownika, otrzymasz wartość NULL, ponieważ wynik działania funkcji SUM (jeśli nie został zwrócony żaden wiersz z powodu braku pracowników) jest równy NULL. Aby pokazać wyniki w bardziej czytelnej postaci, zastąpimy wartość NULL wartością 0, za pomocą funkcji ISNULL:

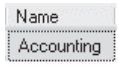
```
SELECT Name, SUM(ISNULL(Salary,0)) as SumaWynagrodzen
FROM Department
LEFT JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Ćwiczenie 4.24

Wyświetl nazwę działu, w którym średnia pensja jest najwyższa.

Oczekiwany wynik

Rysunek 4.27. *Oczekiwany wynik ćwiczenia 4.24*



Rozwiązanie

Połączymy tabele Department i tabele Employee, używając złączenia wewnętrznego INNER JOIN (nie musimy wyświetlać danych tych działów, które nie mają przypisanego żadnego pracownika, bo skoro nie ma w nich pracowników, to wiersze te na pewno nie spełnią warunków zadania — jeśli dział nie ma pracowników, to ich średnia pensja wynosi zero).

Wyniki pogrupujemy według nazwy działu i policzymy średnią pensję dla działu za pomocą funkcji AVG:

```
SELECT Name, AVG(Salary)
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Aby znaleźć dział, w którym są najwyższe zarobki, możemy posortować wyniki malejąco według tych średnich zarobków. W sekcji ORDER BY możemy również używać funkcji grupujących (agregujących):

```
ORDER BY AVG(Salary) DESC
```

Następnie użyjemy klauzuli TOP 1, aby wyświetlić tylko jeden wiersz, znajdujący się na górze — ponieważ posortowaliśmy wyniki malejąco według średnich zarobków, na górze będzie znajdował się wiersz z najwyższymi średnimi zarobkami.

W treści zadania było napisane tylko „wyświetl nazwę działu” — nie było mowy o średnich zarobkach, usuwamy więc z klauzuli SELECT funkcję AVG(Salary).

Ostatecznie zapytanie będzie wyglądało następująco:

```
SELECT TOP 1 Name
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
GROUP BY Name
ORDER BY AVG(Salary) DESC
```

Ćwiczenie 4.25

Policz, ilu pracowników urodziło się w poszczególnych latach.

Oczekiwany wynik

Rysunek 4.28. *Oczekiwany wynik ćwiczenia 4.25*

Rok	LiczbaPracownikow
1978	1
1983	1
1985	1
1987	1
1989	1
1990	2
1992	1
1993	1
1995	1

Rozwiązanie

W klauzuli `GROUP BY` możemy używać nie tylko kolumn, ale również funkcji skalarnych operujących na kolumnach. Możemy więc pogrupować pracowników według roku ich urodzenia, który wyliczymy z daty urodzenia za pomocą funkcji `YEAR`:

```
GROUP BY YEAR(BirthDate)
```

Dla każdej z grup wyświetlamy rok urodzenia oraz liczbę pracowników, którzy urodzili się w tym roku:

```
SELECT YEAR(BirthDate) as Rok, COUNT(*) as LiczbaPracownikow
FROM Employee
GROUP BY YEAR(BirthDate)
```

Pamiętaj, że jeśli grupujesz dane według funkcji skalarnej, której argumentem jest określona kolumna, w klauzuli `SELECT` możesz wyświetlić wartość tej funkcji (tak jak w powyższym wypadku), ale nie możesz próbować wyświetlania tej kolumny, np. poniższe zapytanie zwróci błąd:

```
SELECT BirthDate
FROM Employee
GROUP BY YEAR(BirthDate)
```

Natomiast jeśli grupujesz dane według kolumny, w klauzuli `SELECT` możesz wyświetlić nie tylko tę kolumnę, ale również korzystać z różnych funkcji wyliczających wartość na podstawie tej kolumny, np. poniższe zapytanie jest poprawne:

```
SELECT YEAR(BirthDate)
FROM Employee
GROUP BY BirthDate
```

Ćwiczenie 4.26

Wyświetl wszystkie pierwsze litery imion aktywnych pracowników i policz, ilu pracowników ma imię zaczynające się na daną literę.

Oczekiwany wynik

Rysunek 4.29. *Oczekiwany wynik ćwiczenia 4.26*

PierwszaLitera	LiczbaPracownikow
E	1
J	3
L	1
M	1
P	1

Rozwiązanie

Aby znaleźć pierwszą literę imienia pracownika, użyjemy funkcji `SUBSTRING`, która wyświetla określony podciąg dla danego ciągu znaków. Dla przypomnienia napiszę, że ta funkcja przyjmuje następujące parametry:

- nazwę kolumny, która jest ciągiem znaków,
- numer znaku w ciągu, od którego chcemy rozpocząć dany podciąg znaków,
- liczbę znaków, która będzie długością podciągu.

Pierwszą literę imienia znajdziemy więc w następujący sposób:

```
SUBSTRING(FirstName,1,1)
```

Teraz dla tabeli `Employee`, dla aktywnych pracowników, utworzymy grupy na podstawie pierwszej litery imion i dla każdej grupy policzymy liczbę wierszy:

```
SELECT
   SUBSTRING(FirstName,1,1) as PierwszaLitera,
   COUNT(*) as Liczbapracownikow
FROM Employee
WHERE Active = 1
GROUP BY SUBSTRING(FirstName,1,1)
```

Ćwiczenie 4.27

Dla każdego działu policz, ile pracuje w nim kobiet i ilu mężczyzn.

Oczekiwany wynik

Rysunek 4.30. *Oczekiwany wynik ćwiczenia 4.27*

Name	Gender	LiczbaPracowników
Accounting	M	1
HR	M	2
IT	M	1
Accounting	W	1
IT	W	3
Marketing	W	1

Rozwiązanie

Połączymy tabelę `Department` oraz tabelę `Employee` przy użyciu złączenia wewnętrznego `INNER JOIN`. Następnie otrzymane wyniki pogrupujemy według nazwy działu oraz płci pracownika. Ponieważ nie każdy z pracowników ma wpisaną w bazie danych płeć, użyjemy warunku, który wyświetli tylko tych pracowników, którzy mają podaną płeć. Następnie użyjemy funkcji `COUNT`, aby policzyć liczbę wierszy w każdej grupie.

Zapytanie będzie więc wyglądało następująco:

```
SELECT Name, Gender, COUNT(*) as LiczbaPracowników
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
WHERE Gender IS NOT NULL
GROUP BY Name, Gender
```

Ćwiczenie 4.28

Dla każdego działu znajdź średnią, największą i najmniejszą kwotę pensji.

Oczekiwany wynik

Rysunek 4.31. *Oczekiwany wynik ćwiczenia 4.28*

Name	Średnia pensja	Minimalna pensja	Maksymalna pensja
Accounting	2800.000000	2700.00	2900.00
HR	1250.000000	1000.00	1500.00
IT	1952.500000	1200.00	2500.00
Marketing	1500.000000	1000.00	2000.00

Rozwiązanie

Do tej pory w ćwiczeniach używaliśmy zazwyczaj jednej funkcji grupującej w danym zapytaniu. Nic nie stoi na przeszkodzie, aby użyć ich więcej w jednym zapytaniu i obliczyć więcej wartości dla danej grupy:

```
SELECT
Name,
AVG(Salary) as 'Średnia pensja',
MIN(Salary) as 'Minimalna pensja',
MAX(Salary) as 'Maksymalna pensja'
FROM Employee
INNER JOIN Department ON Employee.DepartmentId = Department.Id
GROUP BY Name
```

Ćwiczenie 4.29

Dla każdego kierownika wyświetl średnie zarobki w jego dziale oraz łączną sumę, którą przeznacza on miesięcznie na wypłaty (oczywiście dla aktywnych pracowników).

Oczekiwany wynik

Rysunek 4.32. *Oczekiwany wynik ćwiczenia 4.29*

FirstName	LastName	Name	Średnie zarobki	Suma wypłat
James	Wilson	HR	1250.000000	2500.00
Johnatan	Davis	IT	1952.500000	7810.00
Linda	Brown	Accounting	2700.000000	2700.00

Rozwiązanie

Aby znaleźć dane kierowników, połączymy tabelę `Department` z tabelą `Employee` na podstawie warunku złączenia `ON Manager.Id = Department.ManagerId`. Dodatkowo po raz kolejny dołączymy tabelę `Employee` — żeby znaleźć wszystkich aktywnych pracowników tego działu. Użyjemy tu więc innego warunku złączenia `ON Department.Id = Employee.DepartmentId`. Pamiętamy o tym, aby nadać aliasy tabeli `Employee`, ponieważ używamy jej dwa razy w tym samym zapytaniu.

Następnie zbudujemy grupę w oparciu o dane kierownika (imię i nazwisko) oraz nazwę działu. Dla każdej z tych grup policzymy średnią, minimalną i maksymalną wysokość zarobków.

Zapytanie będzie wyglądało następująco:

```
SELECT
Manager.FirstName,
Manager.LastName,
Department.Name,
AVG(Employee.Salary) as 'średnie zarobki',
SUM(Employee.Salary) as 'Suma wypłat'
FROM Employee as Manager
INNER JOIN Department ON Manager.Id = Department.ManagerId
INNER JOIN Employee ON Department.Id = Employee.DepartmentId
WHERE Active = 1
GROUP BY Manager.FirstName, Manager.LastName, Department.Name
```

Ćwiczenie 4.30

Oblicz wysokość średniej płacy aktywnych pracowników w działach. Wyniki ogranicz tylko do tych działów, w których średnie zarobki przekraczają 1500 zł. Wyniki posortuj według nazwy działu.

Oczekiwany wynik

Rysunek 4.33. *Oczekiwany wynik ćwiczenia 4.30*

Name	Srednia_pensja
Accounting	2700.000000
IT	1952.500000

Rozwiązanie

Aby znaleźć wysokość średniej płacy aktywnych pracowników w poszczególnych działach, napiszemy zapytanie:

```
SELECT Name, AVG(Salary) as Srednia_pensja
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
WHERE Active = 1
GROUP BY Name
```

Chcemy jednak ograniczyć te wyniki tylko do tych działów, w których średnia pensja jest większa niż 1500 zł. Musimy więc porównać tę wartość ze średnią pensją w każdej grupie (w tym wypadku: w każdym dziale).

Aby rozdzielić warunki nakładane na wiersze przed pogrupowaniem (sekcja `WHERE`) od warunków nakładanych na grupę, została utworzona nowa sekcja — `HAVING`. W tej sekcji mamy dostępne tylko kolumny grup, czyli te kolumny, które zostały użyte do grupowania, oraz funkcje agregujące (grupujące). Sekcja `HAVING` występuje w zapytaniu po sekcji `GROUP BY`. Ostatnią sekcją zapytania jest sekcja `ORDER BY`.

Zapytanie będzie więc wyglądało następująco:

```
SELECT Name, AVG(Salary) as Srednia_pensja
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
WHERE Active = 1
GROUP BY Name
HAVING AVG(Salary) > 1500
ORDER BY Name
```

Ponieważ podczas przetwarzania zapytania sekcja `HAVING` jest wykonywana przed sekcją `SELECT`, nie możemy w sekcji `HAVING` użyć aliasu, który został nadany w sekcji `SELECT`. Alias ten w tym momencie nie jest jeszcze znany. Próba wykonania poniższego zapytania zakończyłaby się więc błędem:

```
SELECT AVG(Salary) as Srednia_pensja
FROM Employee
GROUP BY LastName
HAVING Srednia_pensja > 1500
```

Ćwiczenie 4.31

Wyświetl miasta, w których znajduje się więcej niż jeden dział.

Oczekiwany wynik

Rysunek 4.34. *Oczekiwany wynik ćwiczenia 4.31*

Rozwiązanie

Aby otrzymać informację, w jakim mieście znajduje się dany dział, musimy połączyć tabelę `Department` z tabelą `City`. Utworzymy grupy według nazwy miasta i tak zdefiniujemy dla nich warunek, aby zostały wyświetlone tylko te grupy, które mają więcej niż jeden wiersz:

```
SELECT City.Name
FROM City
INNER JOIN Department ON Department.CityId = City.Id
GROUP BY City.Name
HAVING COUNT(*) > 1
```

Ćwiczenie 4.32

Policz średnią wieku aktywnych pracowników w poszczególnych działach. Wyświetl tylko te działy, w których średnia wieku jest większa niż 27 lat.

Oczekiwany wynik

Rysunek 4.35. *Oczekiwany wynik ćwiczenia 4.32*

Name	Sredni_wiek
Accounting	32
IT	28

Rozwiązanie

Aby policzyć wiek pracownika, użyjemy funkcji `DATEDIFF`, która policzy liczbę lat od daty urodzenia pracownika do dnia dzisiejszego. Połączymy tabelę `Employee` z tabelą `Department` i pogrupujemy wyniki według nazwy działu i obliczymy średnią wieku.

W sekcji `HAVING` możemy również używać różnych obliczeń, możemy więc użyć funkcji `DATEDIFF` (która będzie parametrem funkcji agregującej `AVG`), aby wyświetlić tylko te grupy, gdzie średni wiek pracownika jest wyższy niż 30 lat.

Zapytanie będzie wyglądać następująco:

```
SELECT Name, AVG(DATEDIFF(year,BirthDate,GETDATE())) as Sredni_wiek
FROM Department
INNER JOIN Employee ON Employee.DepartmentId = Department.Id
WHERE Active = 1
GROUP BY Name
HAVING AVG(DATEDIFF(year,BirthDate,GETDATE())) > 30
```

Zadania do samodzielnego wykonania

Zadanie 4.1

Policz wszystkie towary z tabeli `Product`.

Oczekiwany wynik

Rysunek 4.36.
Oczekiwany wynik zadania 4.1

LiczbaTowarow
5

Zadanie 4.2

Policz wszystkie towary, których sugerowana cena (kolumna `Price`) jest niższa niż 100 zł.

Oczekiwany wynik

Rysunek 4.37.
Oczekiwany wynik zadania 4.2

LiczbaTowarow
1

Zadanie 4.3

Policz wszystkie zamówienia (tabela `Orders`) złożone w 2015 roku (kolumna `OrderDate`).

Oczekiwany wynik

Rysunek 4.38.
Oczekiwany wynik zadania 4.3

LiczbaZamowien
4

Zadanie 4.4

Znajdź średnią oraz najniższą i najwyższą sugerowaną cenę spośród wszystkich towarów.

Oczekiwany wynik

Rysunek 4.39.
Oczekiwany wynik zadania 4.4

SredniaCena	NajnizszaCena	NajwyzszaCena
570.000000	50	2000

Zadanie 4.5

Wyświetl średnią cenę produktów z kategorii `Drive`.

Oczekiwany wynik

Rysunek 4.40. *Oczekiwany wynik zadania 4.5*

SredniaCena
250.000000

Zadanie 4.6

Wyświetl łączną sumę wartości wszystkich zamówień złożonych w lutym 2015 roku.

Oczekiwany wynik

Rysunek 4.41. *Oczekiwany wynik zadania 4.6*

SumaZamowien
668

Zadanie 4.7

Policz sumę wartości zamówień wystawionych przez kobiety.

Oczekiwany wynik

Rysunek 4.42. *Oczekiwany wynik zadania 4.7*

SumaZamowien
6445

Zadanie 4.8

Wyświetl nazwy wszystkich grup produktów oraz liczbę produktów znajdujących się w danej grupie.

Oczekiwany wynik

Rysunek 4.43. *Oczekiwany wynik zadania 4.8*

Name	LiczbaProduktow
Accessory	2
Computer	1
Drive	2
Printer	0

Zadanie 4.9

Dla każdego pracownika wyświetl sumę wartości złożonych przez niego zamówień. W wynikach nie uwzględniaj pracowników, którzy nie złożyli żadnego zamówienia.

Oczekiwany wynik

Rysunek 4.44. *Oczekiwany wynik zadania 4.9*

LastName	FirstName	WartoscZamowien
Davis	Johnatan	668
Brown	Linda	2700
Smith	Mary	2890
Wilson	Patricia	855

Zadanie 4.10

Wyświetl imię i nazwisko pracownika oraz sumę złożonych przez niego zamówień w 2015 roku. Wyniki ogranicz tylko do tych pracowników, dla których ta suma zamówień wynosi pomiędzy 2000 zł i 3000 zł.

Oczekiwany wynik

Rysunek 4.45. *Oczekiwany wynik zadania 4.10*

LastName	FirstName	WartoscZamowien
Brown	Linda	2700
Smith	Mary	2890

Zadanie 4.11

Dla każdego towaru wyświetl łączną liczbę tych z nich, które zostały zamówione.

Oczekiwany wynik

Rysunek 4.46. *Oczekiwany wynik zadania 4.11*

Name	ZamowionaLiczba
Flash drive	2
Hard drive	1
Keyboard	0
Laptop	2
Mouse	24

Zadanie 4.12

Dla każdego produktu, który był kiedykolwiek zamówiony, wyświetl datę, kiedy stało się to po raz pierwszy.

Skorowidz

★SQL★ JAK OSIĄGNĄĆ MISTRZOSTWO W KONSTRUOWANIU ZAPYTAŃ

Zaprzyjaźnij się z SQL-em!

- Proste zapytania i sortowanie wyników, czyli od czego zacząć poznawanie składni SQL-a
- Funkcje i operacje, czyli jak ułatwić sobie komunikację z bazą danych
- Zadania, czyli jak zdobyć i rzetelnie przetestować swoje umiejętności

SQL, podstawowy język służący do komunikowania się z bazami danych, na pierwszy rzut oka nie wydaje się zbyt trudny. Ma przejrzystą składnię i sporo pomocnych funkcji, a ponadto jest elastyczny. Jest tylko jeden warunek: żeby sprawnie, szybko wyszukiwać i wyświetlać informacje z bazy danych, musimy go dobrze opanować. Nie chodzi tu tylko o samą konstrukcję zapytań, a raczej o sensowne wyłuskiwanie żądanej informacji spośród tysięcy innych, być może całkiem podobnych. Chodzi o to, by być przygotowanym na nietypowe sytuacje, umieć skonstruować skomplikowane zapytanie z wieloma warunkami i odpowiednio je doprecyzować.

W tej książce znajdziesz setki ćwiczeń i zadań do samodzielnego wykonania. Wszystkie one mają jeden cel: przygotować Cię do wszechstronnej komunikacji z bazą danych i nauczyć Cię wysyłania nawet najbardziej złożonych zapytań. W każdym rozdziale znalazło się nieco teorii i mnóstwo praktyki, a zadania w kolejnych częściach wymagają wiedzy z poprzednich, co pozwala na ciągłe utrwalanie zdobytych wiadomości. Jeśli chcesz rzetelnie i od podszewki poznać bogactwo SQL-a, ta książka z pewnością Ci w tym pomoże!

Zacznij rozmawiać z własną bazą danych!

- Klauzula SELECT
- Funkcje wbudowane
- Złączenia
- Funkcje agregujące
- Operacje na zbiorach
- Podzapytania
- CASE, CAST i CONVERT
- Zadania

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-283-1283-8

9 788328 312838

Informatyka w najlepszym wydaniu

cena: 54,90 zł